Neutrosophic Sets and Systems, Vol. 76, 2025
University of New Mexico

NSDTL: A Robust Malware Detection Framework Under Uncertainty

Alaa Elmor [1]
[1] Zagazig University, 44519 Zagazig, Egypt
Email: alaaelmor@zu.edu.eg
* Correspondence: alaaelmor@zu.edu.eg

Abstract: The Internet's rapid expansion and the current trends toward automation through intelligent systems have given malevolent software attackers a veritable playground. Numerous gadgets are effortlessly connected to the Internet, and a lot of data is being collected. Consequently, there is a growing concern about malware attacks and security threats. Malware detection has emerged as a research focus. However, there are challenges in the research, such as noise, uncertainty, and ambiguous data. The study proposes a novel framework NSDTL, that achieves state-of-the-art malware detection and classification results to address this changing threat landscape. NSDTL leverages a neutrosophic set and advanced transfer learning techniques. There are three different kinds of images in the neutrosophic domain: True (T) images, Indeterminacy (I) images, and Falsity (F) images, which deal with uncertainty. The MaleVis dataset was used for experiments on multi-class malware classification, and the findings show that NSDTL significantly outperforms current models. This study emphasizes how crucial it is to combine transfer learning with a neutrosophic set at the forefront of the continuous fight against changing cyber threats.

Keywords: Neutrosophic Set; Deep Transfer Learning; Malware detection; IoT.

1. Introduction

Now, we enter the fifth industrial revolution (Industry 5.0), which is being propelled by developments in Internet technologies and intelligent automation, we are encountering more and more security breaches. Developing defenses against malicious software assaults is crucial because these attacks have the potential to damage enterprises by infiltrating their data, applications, and computer systems without obtaining user authorization or authentication [1, 2]. Scholars, businesses, financial organizations, and governments are very concerned about these concerns, which include file loss, hostile extortion, information theft, telecommunication fraud, firm shutdowns, and other problems [3]. The two primary kinds of malware detection techniques are traditional static detection techniques [4] and dynamic detection techniques [5, 6]. The static technique determines whether the software is malware by examining its structure. The dynamic technique determines whether running software is malware by examining its behavior. Despite its extremely high accuracy rate, the dynamic detection technique requires program monitoring, which takes time and makes it difficult to discover dangerous software at the moment. Although static detection can compensate for this shortcoming, traditional static detection techniques depend on robust virus databases and effective anti-virus engines. As a result, it is exceedingly challenging to identify invisible malware, which contributes to the subpar effectiveness of traditional static detection techniques [7]. Some researchers proposed several techniques to identify malware by utilizing visualization technologies to make up for the drawbacks of traditional static detection [8, 9]. It has been demonstrated that these techniques function well in the area of malware detection. The binary code or assembly code of the majority of malware versions is quite similar since they are frequently created by automation technology or by reusing key function modules [10]], this can be seen in the visual image and makes visualization technology effective. Transfer learning-based malware classification has been examined in several previous literary works. Reputable CNN architectures can be used to refine a classification model for the target domain [11] or as feature extractors with machine learning algorithms for classification on the target domain [12-14]. Nevertheless, not many studies have yet to tackle uncertainty issues.

To address higher degrees of uncertainty, Smarandache [15] introduced a neutrosophic set (NS) in 1998. Since then, NS has been applied to a wide range of computer science domains, such as image processing and segmentation [16], pattern recognition [17], and more. In various fields, including medical [18], space satellites [19], and agriculture[20], it helps to solve several research and real-world problems. Three distinct degrees of membership—true, indeterminacy, and false—can be assigned by NS to any element in the universe. Incomplete, inconsistent, and ambiguous information—which is found in malware images—can be represented using these domains.

In this study, we introduce a novel framework that integrates DTL and NS to handle uncertainty in three degrees of membership for malware detection. The frameworks employ four different DTL models, VGG, MobileNetV2, DenseNet121, and Xception. We conduct our study in the MaleVis dataset, we concentrate on identifying different malware with high accuracy, in contrast to previous studies focusing on binary classification for high accuracy but may neglect the type of malware. VGG16 shows the highest results in terms of Accuracy, MCC, Precision, Recall, F1-score, and AUC with 93.63%, 93.35%, 95.29%, 93.6%, 94.27%, and 97% respectively in the true domain.

The rest of the paper is organized as follows: Section 2 covers the related work of this study. Section 3 describes the proposed framework, including the NS-based data preprocessing technique and utilized DTL models. Section 4 presents the experimental setup and describes the MaleVis dataset. Section 5 discusses the experiments and findings. Section 6 concludes the study and discusses future directions. Finally, Section 6 presents the new directions of neutrosophic logic in Malware Detection.

1.1 The Role of Neutrosophic Logic in Malware Detection

Neutrosophic logic (NL) extends traditional binary logic (true or false) by introducing a third state, which represents indeterminacy. This makes it suitable for scenarios where information is incomplete or uncertain, which is common in cybersecurity. NL is a mathematical framework that allows for the representation of uncertain, indeterminate, or contradictory information. In the context of cybersecurity, particularly in malware detection, neutrosophic logic provides significant advantages in handling the complexities and ambiguities inherent in identifying malicious software.

How Neutrosophic Logic Applied in Malware Detection

- Malware detection often relies on various indicators, such as file behavior or network activity, which may be incomplete or ambiguous. NL can effectively analyze these uncertain data points to provide a more accurate assessment of whether a file or process is malicious.

- NL allows for the development of dynamic classification systems that can categorize different types of malware based on their features, even when these features may conflict or be poorly defined.
- By incorporating neutrosophic logic into machine learning algorithms, it is possible to improve the robustness of models used for malware detection. This integration helps the models adapt to changing data patterns and increases their accuracy.
- A common challenge in malware detection is the occurrence of false positives, where legitimate software is mistakenly identified as malware. NL helps mitigate this issue by providing a nuanced analysis that considers degrees of uncertainty, leading to more reliable detection outcomes.

1.2 Applications of Neutrosophic in Malware Detection

1- In malware detection, various indicators such as file signatures, behavior patterns, and network traffic can often be uncertain. By applying neutrosophic logic, we can represent these indicators as neutrosophic triples. For example, if a certain file is suspected of being malware, we might assess:

$$\text{File\_x} = (0.7,0.2,0.1)$$

This indicates a 70% chance that it is malware, a 20% uncertainty, and a 10% chance that it is definitely not malware.

2- When analyzing multiple files, neutrosophic logic allows security systems to aggregate information from various sources. If we have multiple files *File_x* and *File_y*, their combined representation can be evaluated to determine if the overall risk increases or decreases.

3- Traditional detection systems often suffer from high false positive rates. By integrating neutrosophic logic, these systems can more accurately assess the likelihood of a file being malicious by considering the uncertainty and providing a more balanced view. For example:

$$\text{Risk} = (T_{\text{malware}}, I_{\text{uncertainty}}, F_{\text{legitimate}})$$

4- Neutrosophic logic can enhance machine learning models by providing a framework that can adapt as new information becomes available. As more data is collected, the neutrosophic values can be updated, improving the model's ability to detect new types of malware.

The incorporation of neutrosophic logic into existing detection frameworks represents a significant advancement in the fight against malware, providing deeper insights and improved decision-making capabilities.

2. Related work

This section provides a summary of research on visualization-based methods for classifying malware. Code obfuscation or encoding problems can be resolved by image-based malware classification techniques [21-25]. These image-based methods convert malware samples into image representations using static analysis techniques. These representations are then fed into image classification methods to determine if the sample is benign or malicious.

[26] Jadeite, a method for identifying Java bytecode malware programs, is presented by Obaidat et al. From a Java bytecode file, Jadeite extracts the Interprocedural Control Flow Graph (ICFG), prunes

it, and transforms it into an adjacency matrix. Jadeite then uses this matrix to create a grayscale image. To increase the precision of malware classification, Jadeite also extracts a different set of features from the Java malware program. The CNN classifier uses these features as inputs combined with the retrieved images.

The malware detection architecture for the Industrial Internet of Things (MD-IIOT) was designed by Naeem et al. [27], they set up three databases—traffic, behavior, and log databases—in cloud storage. The traffic database uses a sniffing and monitoring unit to gather the IIOT's raw Android files. A list of datasets representing the network history makes up a behavior database. The log database continually updates the behavior database and gathers new malware fingerprints. A Deep Convolutional Network (DCNN) and a color picture modification approach are designed for the analyzer unit. A DCNN model is applied once the raw Android file has been transformed into a color image. The analyzer unit assesses the file as benign or malicious after comparing its fingerprints with a behavior database. The response unit sends the system administrator a last warning to take appropriate action if any malicious behavior is detected within the network.

Ding et al. [28] utilize CNN to train a malware detection model after directly extracting the bytecode file from the Android APK file and converting it into a two-dimensional bytecode matrix. The experimental findings demonstrate the efficacy of the suggested approach in detecting malware, particularly malware that has been encrypted via polymorphic approaches.

A temporal convolution network (TCN)--based malware detection model for Android was presented by Zhang et al. [29]. First, XML files and DEX files are combined to create four grayscale image datasets with four distinct combinations of texture features. For experimental validation, the image size is then combined and fed into the neural network design using three distinct convolution techniques. According to the testing findings, adding XML files improves Android malware detection.

A technique to determine if generative adversarial network-generated images can be differentiated from actual images was presented by Mercaldo et al. [30] using a dataset of real-world Android malware programs. Using images from both static analysis and dynamic analysis, their trials included two different kinds of deep convolutional generative adversarial networks. They trained several supervised machine learning models after creating images to see if they could distinguish between harmful applications that were produced and those that were real.

A malware classification approach was presented by Ünver et al. [31] to identify malware samples in the Android environment. The base of the proposed model is the conversion of some Android application source files into grayscale images. The proposed model has been trained using a few image-based local and global features, four distinct local feature types, and three distinct global feature types, which have been taken from the created grayscale image datasets. One feature vector was created using the bag of visual words approach using the descriptors of the local features that were taken out of each image. Several machine learning classifiers, including Random Forest, k-nearest neighbors, Decision Tree, Bagging, AdaBoost, and Gradient Boost, have been trained using the extracted local and global features.

To classify 25 malware families using images, both with and without class balancing, Awan et al. [32] propose a deep learning framework based on spatial attention and CNN. Precision, recall, specificity, precision, and F1 score were used to assess performance on the Malimg dataset. The proposed model with class balance achieved 97.42%, 97.95%, 97.33%, 97.11%, and 97.32% on this dataset and with class balancing on a benign class, which yielded results above 97%.

Shaukat et al. [33] proposed a malware detection approach based on deep learning, A portable executable (PE) file is first visualized as a colored image. Second, it uses a refined deep learning model to extract deep information from the color image. Third, it uses support vector machines (SVM) to identify malware based on deep features.

An automatic extraction technique without the need for human-expert interaction was presented by Zhu et al. [34]. They compare an RGB image to the essential components of the Dalvik executable (Dex). By utilizing multi-scale context information, they introduced MADRF-CNN, a CNN variation with heterogeneous receptive fields that uses max pooling and average pooling simultaneously to capture the dependencies between various picture elements (transferred from the Dex file). According to the experimental findings, their method's accuracy is 96.9%.

A snake optimization algorithm with a deep convolutional neural network for an image-based malware classification approach is presented by Duraibi et al. [35]. The proposed approach attempts to identify malware images using a hyperparameter-tuned deep learning approach. The feature vectors are primarily derived using the ShuffleNet approach. Additionally, the snake optimization technique can be used to improve the ShuffleNet algorithm's hyperparameter selection. A bi-directional long short-term memory model based on attention is used to identify malware images. The Malimg malware dataset has been used to evaluate the proposed approach. According to the experimental results, the proposed approach yields a 98.42% accuracy rate.

The **neutrosophic set** has been used in cybersecurity research in several areas, such as intrusion detection and risk assessment. Awotunde et al. [36] developed a multilevel random forest algorithm for intrusion detection, using fuzzy logic, intrusions were categorized as normal, low, medium, or high. Also, Ap et al. [37] presented a lightweight intrusion detection system based on fuzzy logic to identify malicious activities occurring during inter-IoT device connection. In risk assessment, Abdel-Basset et al. [38] proposed the RAF-CPWS methodology using neutrosophic theory to assess risks in wastewater treatment technologies, considering several social, technological, environmental, economic, and cybersecurity factors. Another study by Abdel-Basset et al. [39] combined neutrosophic sets with MCDM techniques for assessing security risks in self-driving cars.

However, this is the first study to apply the neutrosophic domain to malware images, making it a novel contribution to the field.

3. Methodology

Neutrosophic domain conversion and transfer learning architectures are the two primary components of the technique framework suggested in this study to identify 25 distinct kinds of malware. The architecture of the proposed NDDTL framework is shown in Figure 1. As a preprocessing step, the neutrosophic image conversion is applied, and as a classification model, the DTL model is employed.

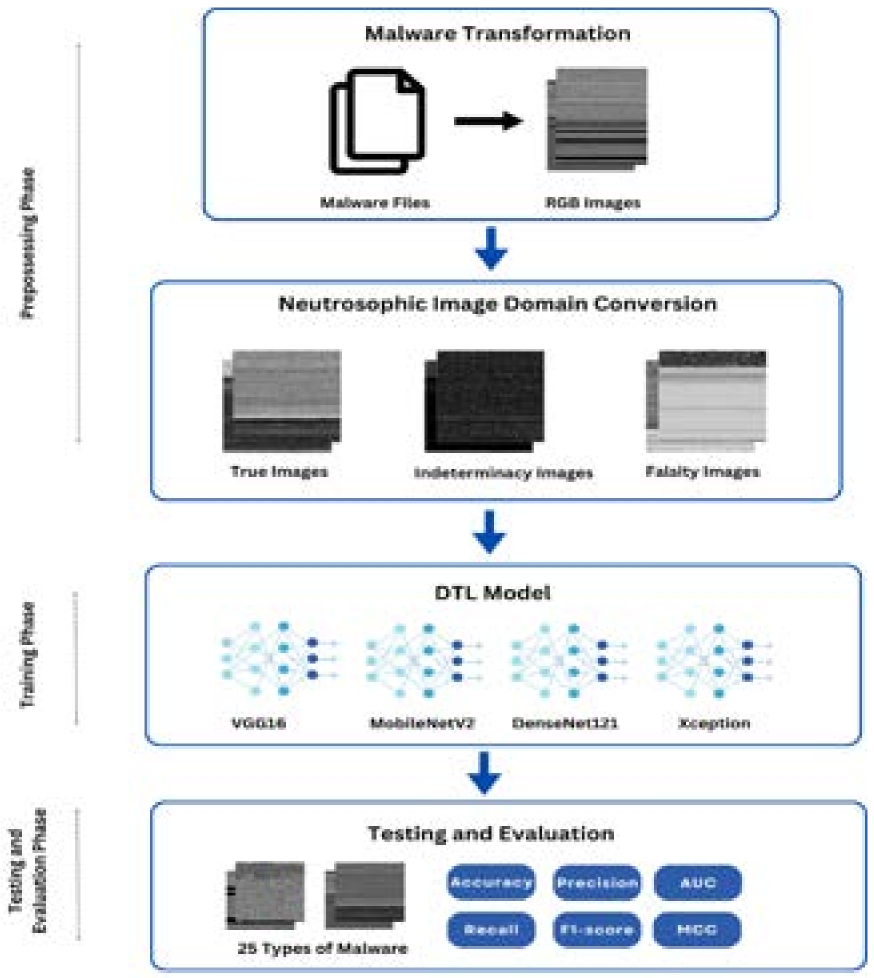


Fig.1 The detailed architecture of proposed NSDTL framework

3.1. Neutrosophic Image Domain Conversion

A practical and beneficial theory for computing fuzzy circumstances is a neutrosophic set (NS). According to NS theory, events are calculated by subtracting them into three categories: true (T) significance, where the status is a percentage of true; indeterminacy (I) significance, where the status is a percentage of indefinite; and falsity (F) significance, where the status is a percentage of false. When classifying malware images, each pixel is separated into T, I, and F subsets. These subsets are then used to classify the image. As demonstrated by Eqs. 1–7, the input image transforms into the neutrosophic domain. The image domain P(x) pixel is transformed into the neutrosophic domain $P_{NS}(x)$:

$$P_{NS}(x) = \{T(x),\ I(x),\ F(x)\} \quad (1)$$

Using a convolution process, the local mean g(x,y) of an image G can be used to convert it into the domains of a malware image G with pixel intensities. Each pixel in the image is represented as P(x,y). Next, Determine the absolute difference between the image's mean O(x,y) and the original image.

$$g(x,y) = \frac{1}{25}\sum_{i=-2}^{2}\sum_{j=-2}^{2} G(x+i, y+j) \quad (2)$$

$$O(x, y) = abs(G(x, y) - g(x, y)) \tag{3}$$

The NS domains for the pixel P(x,y) can therefore be shown and delivered as follows:

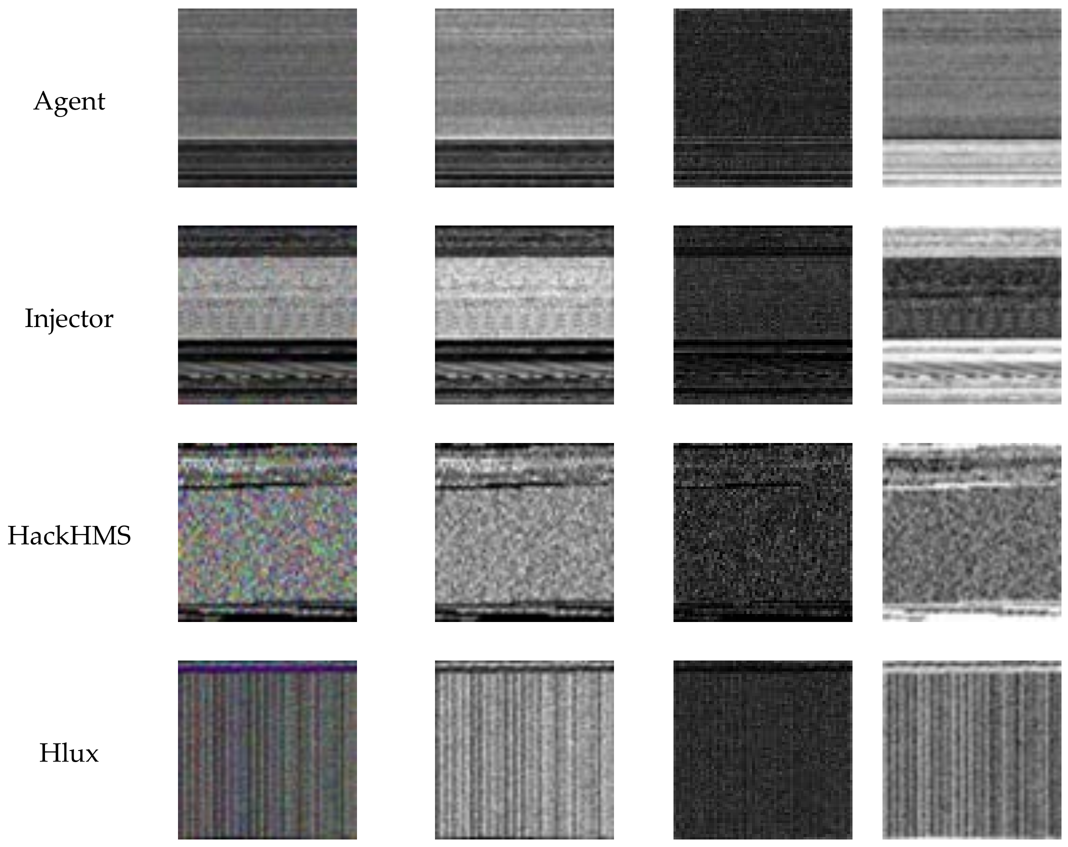
$$P_{NS}(x, y) = \{T(x, y), I(x, y), T(x, y)\} \tag{4}$$

$$T(x, y) = \frac{g(x, y) - g_{min}}{g_{max} - g_{min}} \tag{5}$$

$$I(x, y) = \frac{O(x, y) - O_{min}}{O_{max} - O_{min}} \tag{6}$$

$$F(x, y) = 1 - T(x, y) \tag{7}$$

G(x,y) denotes the pixel value at the coordinate (x,y) in the image, where x and y variables normally indicate the pixel coordinates in a 2D image. The neighboring pixels surrounding a given pixel (x,y) are indicated by the indices G, I, and j. The highest average pixel value in a specified neighborhood is known as g_max. The lowest average pixel value in the same neighborhood is called g_min. The greatest absolute difference seen throughout the neighborhood is known as O_max. The smallest absolute difference that has been observed is O_min [40,41]. Figure 2 shows samples of malware images for five different classes (out of the dataset's twenty-five classes) after the conversion of the neutrosophic image domain in various domains.



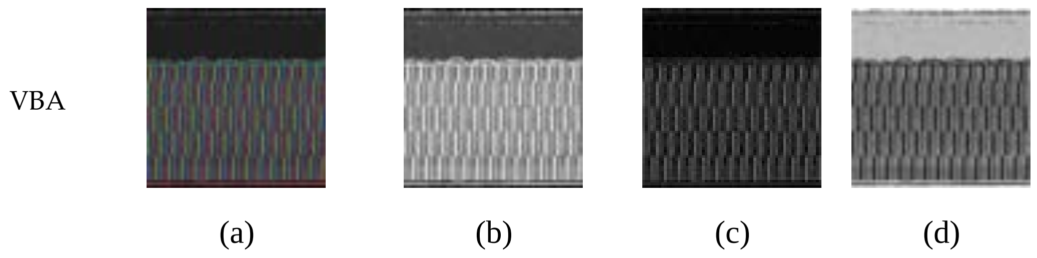


Fig. 2 Five malware families using different neutrosophic image domains (a) original images, (b) true significance images, (b) indeterminacy significance images, and (d) falsity significance images

3.2. Deep Transfer Learning Model

CNNs are ideal for image classification because they gradually extract a more meaningful representation of the visual content. CNN uses the entire image as input, creates useful feature maps from the image, and then deduces information about the item it contains, as opposed to processing the data to extract an image's features. It takes a lot of time and depends on the dataset to develop and train a new CNN model. Transfer learning is helpful in this situation since it offers a technique that allows knowledge to be transferred from the source domain to the target domain with minimal adjustments to the deep CNN models [42]. Four DTL models—VGG16[43], MobileNetV2[44], DenseNet121[45], and Xception[46]—are used in this study to classify malware families. Convolutional layers from the TL model, which was previously trained on the ImageNet dataset, are transferred to our DNN model to construct a Deep Neural Network (DNN) model. Five convolutional layer blocks make up the first part of the VGG16 architecture, which is followed by three fully connected layers. Utilizing 3 × 3 kernels with a stride of 1 and padding of 1, convolutional layers guarantee that every activation map maintains the same spatial dimensions as the one before it. A max pooling operation is used at the end of each block to minimize the spatial dimension, and a rectified linear unit (ReLU) activation is carried out immediately following each convolution. Each spatial dimension of the activation map from the preceding layer is divided by max-pooling layers using 2 × 2 kernels with a stride of 2 and no padding. Ends with two fully connected layers with 4096 ReLU-activated units and 1000 fully connected softmax. Inverted residual and linear bottleneck modules were added to MobileNetV1 to create MobileNetV2. Depthwise separable convolution served as the foundation for the MobileNet architecture. The conventional 2D convolution convolves in the depth dimension (channel) as well, processing each input channel directly to create one output channel. The input image and filter are divided into distinct channels by the depthwise convolution, which then convolves each input channel with its matching filter channel. Following the production of the filtered output channel, the output channels are stacked back. In inseparable depthwise convolution, the stacked output channels are combined into a single channel by filtering them using a 1×1 convolution, also known as pointwise convolution[47]. A convolutional layer and a connected layer are the two types of layers that make up the DenseNet121 architecture. By tightly connecting layers, densely connected convolutional networks transformed neural network topologies. Every layer receives information from the layers that came before it. By using fewer parameters, this approach allows the network to increase accuracy and reuse features. The term Xception refers to the Extreme version of Inception, which consists of a convolution layer that mixes depthwise and pointwise convolution. This combination was subsequently shown to provide higher classification accuracy for images in a dataset. As the basis of the feature extraction network, the Xception NN layer architecture consists of 36 convolution layers. Except for the first and last modules, all 14 of the 36 convolutional layers are arranged into modules with linear residual connections surrounding them.

4. Experimental Setup

This section covers the datasets and experimental settings used in the study in detail.

4.1. Dataset

The Malevis [48] dataset consists of byte pictures of 26 (25+1) classes and is based on RGB ground truth. In this case, 1 class stands for the "legitimate" samples, whereas the remaining 25 classes represent various malware variants. Using a bin2png script created by Sultanik, the binary images from malware files (provided by Comodo Inc.) were initially extracted in three-channel RGB format to create this dataset. The Malevis dataset consists of 1482 images in total. Malware classes contain Adposhel, Agent, Allaple, Amonetize, Androm, Autorun, BrowseFox, Dinwod, Elex, Expiro, Fasong, HackKMS, Hlux, Injector, InstallCore, MultiPlug, Neoreklami', Neshta, Regrun, Sality, Snarasite, Stantinko, VBA, VBKrypt, and Vilsel.

4.2. Implementation Details

Deep learning frameworks Keras and TensorFlow were used to implement the models. An NVIDIA GeForce RTX 4060 graphics card and an Intel(R) Core (TM) i7-13700HX@2.10 GHz CPU were the hardware components; Windows 11 pro-64-bit was the experimental operating system. The batch size for all models is 64, and there are 30 epochs with early stopping. They employed the Adam optimizer and the Sparse Categorical Cross-Entropy loss function.

4.3. Evaluation Metrics

Several measures of performance, including F1-score, accuracy, recall, precision, and AUC, are provided.

- Accuracy is defined as the proportion of successfully identified instances to all objects in the dataset.

$$ACC = \frac{TP + TN}{TP + FP + FN + TN} \tag{8}$$

- Precision for a particular class is the proportion of examples that are accurately categorized as belonging to a certain class out of all instances of the model that are predicted to belong to that class.

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

- Recall is the proportion of examples in a class that the model correctly identified out of all instances in the class.

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

- The F1 score is a metric that evaluates the model's overall performance by combining recall and precision. The harmonic mean of recall and precision is used to calculate it.

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall} \qquad (11)$$

- MCC measure the discrepancy between the expected and actual values. It is similar to two-by-two contingency table chi-square statistics.

$$\mathrm{MCC} = \frac{\mathrm{TP} * \mathrm{TN} - \mathrm{FP} * \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}} \qquad (12)$$

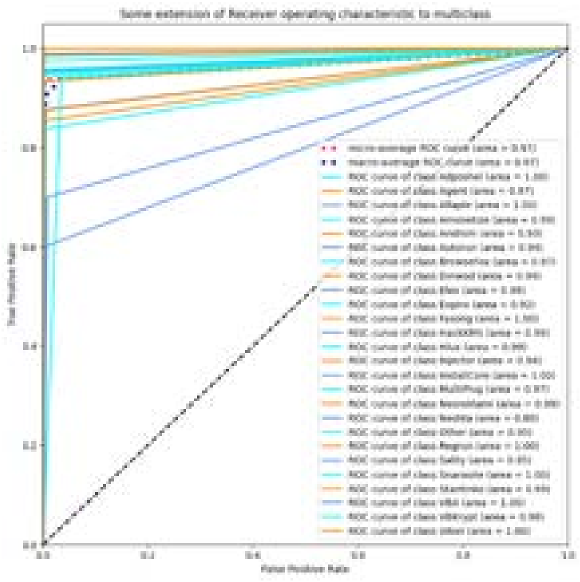
Where TP, FN, TN, and FP stand for the number of true positives, false negatives, and true negatives, and false positives, respectively.

5. Results and Discussion

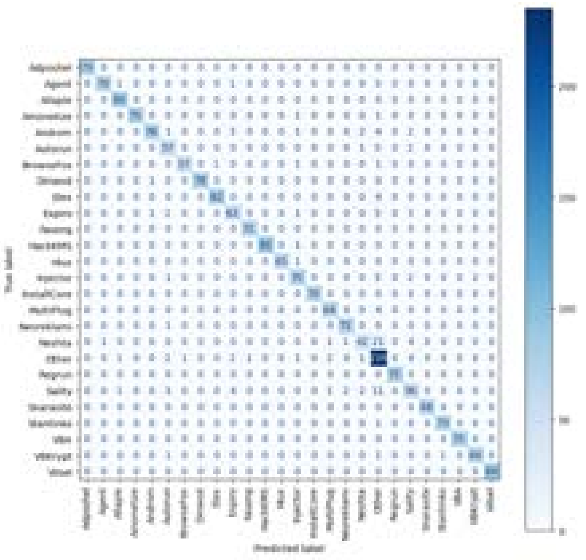
The effectiveness of our malware detection NSDTL framework is discussed in this section. Four distinct DL models—VGG16, MobileNetV2, DenseNet121, and Xception—were used to evaluate our methodology. According to Table 1, deep learning models perform best on the True domain in terms of accuracy, MCC, precision, recall, F1-score, and AUC, with 93.63%, 93.35%, 95.29%, 93.6%, 94.27%, and 97%, respectively. The VGG16 model is superior, according to table 1 in all domains. Figure 3 displays the ROC curves and confusion matrices for DTL models in the True domain.

Table. 1 Comparison of Deep Learning Models Performance on the Original and NS Domains

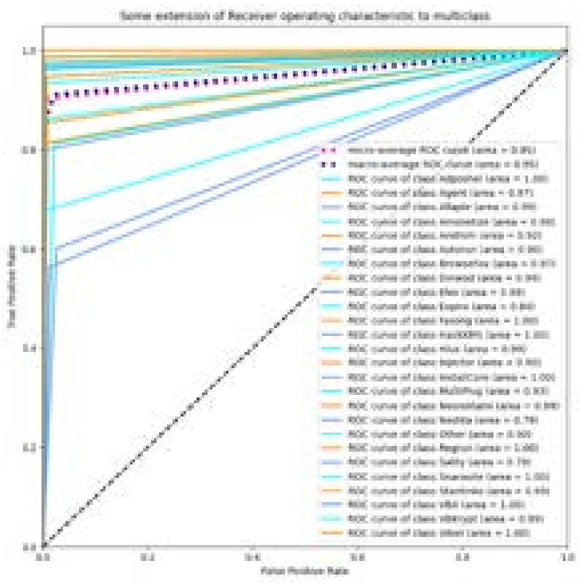
Original Data						
Model	**Accuracy**	**MCC**	**Precision**	**Recall**	**F1-score**	**AUC**
VGG	93.25	92.92	94.45	93.6	93.8	97
MobileNetV2	89.6	89.11	90.52	90.08	90.06	95
DenseNet121	91.56	91.17	93.14	91.95	92.43	96
Xception	87.67	87.11	88.24	88.55	88.26	94
True Domain						
Model	**Accuracy**	**MCC**	**Precision**	**Recall**	**F1-score**	**AUC**
VGG	93.63	93.35	95.29	93.6	94.27	97
MobileNetV2	90.25	89.81	91.38	91.01	91.03	95
DenseNet121	91.28	90.88	92.67	91.68	92.02	96
Xception	88.71	88.19	89.8	89.7	89.61	95
Indeterminacy Domain						
Model	**Accuracy**	**MCC**	**Precision**	**Recall**	**F1-score**	**AUC**
VGG	92.97	92.65	93.86	93.64	93.69	95
MobileNetV2	87.77	87.23	89.36	87.84	88.09	94
DenseNet121	88.57	88.04	89.32	89	88.9	94
Xception	86.03	85.4	87.6	86.84	86.81	93
False Domain						
Model	**Accuracy**	**MCC**	**Precision**	**Recall**	**F1-score**	**AUC**
VGG	93.25	92.94	94.29	93.5	93.75	97
MobileNetV2	89.97	89.5	91.99	90.47	91.03	95
DenseNet121	91.24	90.84	92.2	92.12	91.97	96
Xception	86.31	85.7	88.96	86.39	87.41	93



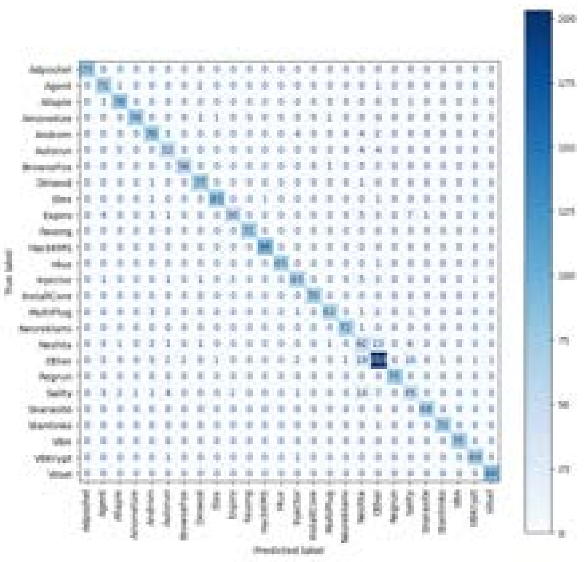
(a) ROC curve of VGG16



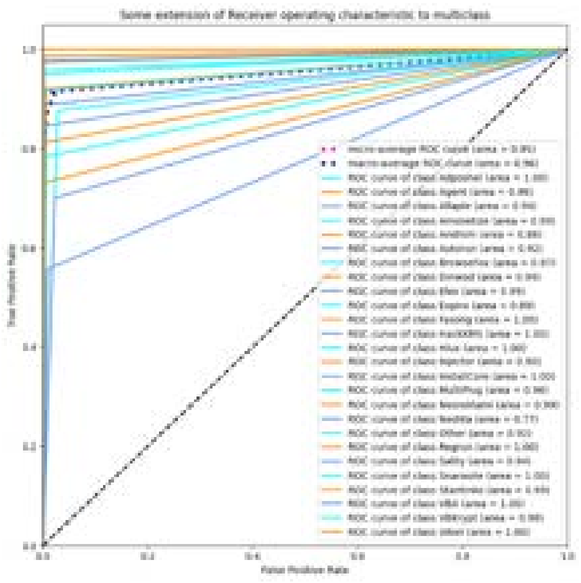
(b) Confusion metrics of VGG16

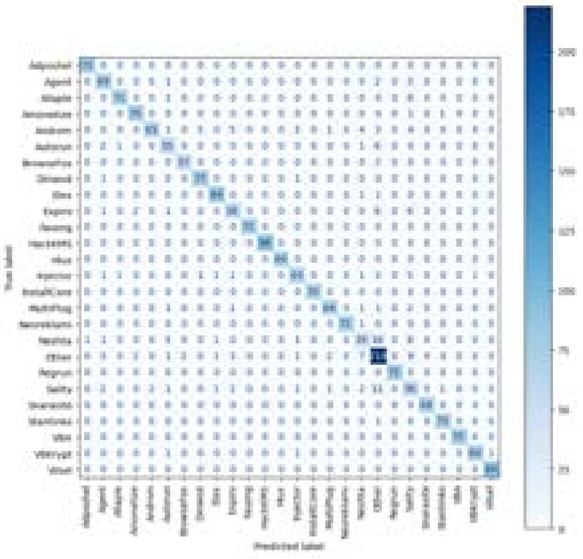


(c) ROC curve of MobileNetV2

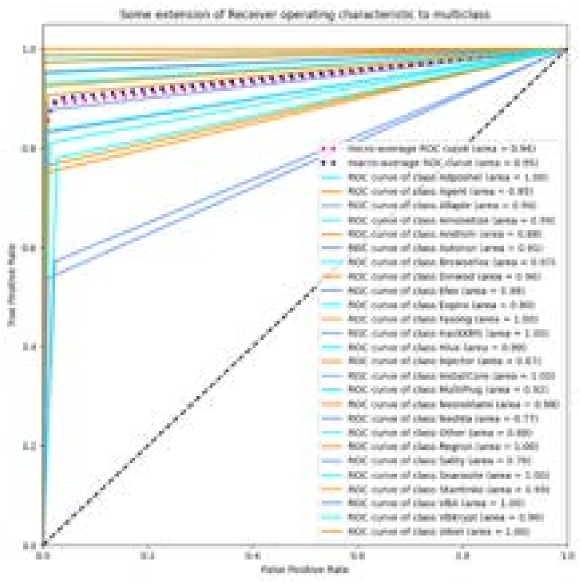


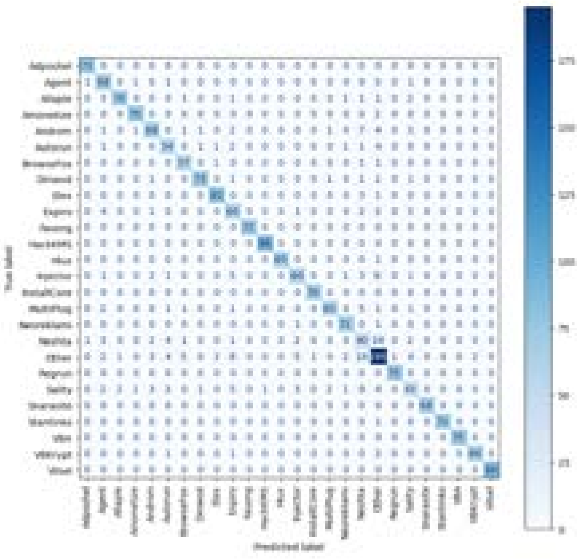
(d) Confusion metrics of MobileNetV2





(e) ROC curve of DenseNet121 (f) Confusion metrics of DenseNet121





(g) (h)

(i) ROC curve of Xception (j) Confusion metrics of Xception

Fig. 3 ROC curves and confusion matrices for DTL models in the True domain

6. Conclusion

In this paper, we integrate the neutrosophic domain with advanced transfer learning algorithms to present a novel NSDTL framework for malware detection. The inherent difficulties presented by ambiguity, noise, and uncertainty in malware datasets are effectively addressed by the approach. The NSDTL framework improves detection capabilities by transforming malware images into the neutrosophic domain, which consists of True, Indeterminacy, and Falsity components. Experiments on the MaleVis dataset show that NSDTL is superior in the True dataset across various evaluation metrics. The best model is the VGG16, which achieves accuracy, MCC, precision, recall, F1-score, and AUC of 93.63%, 93.35%, 95.29%, 93.6%, 94.27%, and 97%, respectively. The findings demonstrate that using the neutrosophic set in conjunction with transfer learning greatly increases the accuracy of malware classification, surpassing current techniques. This study demonstrates how hybrid approaches might improve cybersecurity defenses and provides a promising avenue for further investigation in the continuous fight against evolving cyber threats.

7. New Trends in Neutrosophic Logic for Malware Detection

- Neutrosophic logic allows for the integration of diverse data sources (like behavioral patterns and system logs) into a unified analysis, providing a clearer picture of potential threats.
- By using neutrosophic models, systems can dynamically evaluate the risk levels of files or processes based on varying degrees of uncertainty, leading to more informed decision-making.
- Neutrosophic logic enhances machine learning models by enabling them to adapt to new information and evolving malware techniques, improving detection accuracy over time.
- Neutrosophic methods allow for a more nuanced classification of malware types by considering not just binary classifications (malicious or benign) but a range of possibilities that reflect uncertainty.
- Neutrosophic logic helps minimize false positives by providing a more detailed analysis of uncertainty, allowing legitimate software to be correctly identified without being flagged as malware.
- Neutrosophic approaches are increasingly being applied to analyze the behavior of applications in real time, identifying suspicious activities even when traditional signatures fail.
- Neutrosophic logic supports collaborative frameworks where different systems can share and analyze data collectively, enhancing overall detection capabilities.
- New tools and dashboards using neutrosophic principles allow security analysts to visualize uncertainty and risk levels, making it easier to understand complex threat landscapes.
- Neutrosophic logic is being combined with artificial intelligence and big data analytics to enhance predictive capabilities and improve the overall effectiveness of malware detection systems.
- The principles of neutrosophic logic are being explored in various domains beyond cybersecurity, such as IoT security and cloud computing, demonstrating its versatility and potential for broader application.

These trends highlight how neutrosophic logic is shaping the future of malware detection, offering innovative solutions to the ongoing challenges in cybersecurity.

References

1. Aycock, J., Computer viruses and malware. Vol. 22. 2006: Springer Science & Business Media.
2. Mohamed, G.A.N. and N.B. Ithnin, Survey on representation techniques for malware detection system. Am. J. Appl. Sci, 2017(11): p. 1049-1069.

3. Liu, Y., et al., Efficient and Generalized Image-Based CNN Algorithm for Multi-Class Malware Detection. IEEE Access, 2024.
4. Kruegel, C., et al. Polymorphic worm detection using structural information of executables. in Recent Advances in Intrusion Detection: 8th International Symposium, RAID 2005, Seattle, WA, USA, September 7-9, 2005. Revised Papers 8. 2006. Springer.
5. Huang, W. and J.W. Stokes. MtNet: a multi-task neural network for dynamic malware classification. in Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7-8, 2016, Proceedings 13. 2016. Springer.
6. Lin, C.-H., H.-K. Pao, and J.-W. Liao, Efficient dynamic malware analysis using virtual time control mechanics. Computers & Security, 2018. 73: p. 359-373.
7. Jian, Y., et al., A novel framework for image-based malware detection with a deep neural network. Computers & Security, 2021. 109: p. 102400.
8. Liu, X., et al., A novel method for malware detection on ML-based visualization technique. Computers & Security, 2020. 89: p. 101682.
9. Qiao, Y., et al. A multi-channel visualization method for malware classification based on deep learning. in 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). 2019. IEEE.
10. Han, K.S., et al., Malware analysis using visualized images and entropy graphs. International Journal of Information Security, 2015. 14: p. 1-14.
11. Vasan, D., et al., IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. Computer Networks, 2020. 171: p. 107138.
12. Narayanan, B.N., O. Djaneye-Boundjou, and T.M. Kebede. Performance analysis of machine learning and pattern recognition algorithms for malware classification. in 2016 IEEE national aerospace and electronics conference (NAECON) and ohio innovation summit (OIS). 2016. IEEE.
13. Davuluru, V.S.P., B.N. Narayanan, and E.J. Balster. Convolutional neural networks as classification tools and feature extractors for distinguishing malware programs. in 2019 IEEE National Aerospace and Electronics Conference (NAECON). 2019. IEEE.
14. Narayanan, B.N. and V.S.P. Davuluru, Ensemble malware classification system using deep neural networks. Electronics, 2020. 9(5): p. 721.
15. Smarandache, F., et al., Introduction to neutrosophy and neutrosophic environment, in Neutrosophic Set in Medical Image Analysis. 2019, Elsevier. p. 3-29.
16. Salama, A., Basic structure of some classes of neutrosophic crisp nearly open sets and possible application to GIS topology. Neutrosophic Sets and Systems, 2015. 7: p. 18-22.
17. Ali, M., I. Deli, and F. Smarandache, The theory of neutrosophic cubic sets and their applications in pattern recognition. Journal of intelligent & fuzzy systems, 2016. 30(4): p. 1957-1963.
18. Guo, Y. and A.S. Ashour, Neutrosophic set in medical image analysis. 2019: Academic Press.
19. Bausys, R., et al., Algorithm selection for edge detection in satellite images by neutrosophic WASPAS method. Sustainability, 2020. 12(2): p. 548.

20. Dhingra, G., V. Kumar, and H.D. Joshi, A novel computer vision based neutrosophic approach for leaf disease identification and classification. Measurement, 2019. 135: p. 782-794.
21. Hashemi, H. and A. Hamzeh, Visual malware detection using local malicious pattern. Journal of Computer Virology and Hacking Techniques, 2019. 15: p. 1-14.
22. Fu, J., et al., Malware visualization for fine-grained classification. IEEE Access, 2018. 6: p. 14510-14523.
23. Le, Q., et al., Deep learning at the shallow end: Malware classification for non-domain experts. Digital Investigation, 2018. 26: p. S118-S126.
24. Vu, D.L., et al., HIT4Mal: Hybrid image transformation for malware classification. Transactions on Emerging Telecommunications Technologies, 2020. 31(11): p. e3789.
25. Efe, A. and S.H.S. Hussin, Malware visualization techniques. International Journal of Applied Mathematics Electronics and Computers, 2020. 8(1): p. 7-20.
26. Obaidat, I., et al., Jadeite: a novel image-behavior-based approach for java malware detection using deep learning. Computers & Security, 2022. 113: p. 102547.
27. Naeem, H., et al., Malware detection in industrial internet of things based on hybrid image visualization and deep learning model. Ad Hoc Networks, 2020. 105: p. 102154.
28. Ding, Y., et al., Android malware detection method based on bytecode image. Journal of Ambient Intelligence and Humanized Computing, 2023. 14(5): p. 6401-6410.
29. Zhang, W., et al., Android malware detection using tcn with bytecode image. Symmetry, 2021. 13(7): p. 1107.
30. Mercaldo, F., F. Martinelli, and A. Santone, Deep Convolutional Generative Adversarial Networks in Image-Based Android Malware Detection. Computers, 2024. 13(6): p. 154.
31. Ünver, H.M. and K. Bakour, Android malware detection based on image-based features and machine learning techniques. SN Applied Sciences, 2020. 2(7): p. 1299.
32. Awan, M.J., et al., Image-based malware classification using VGG19 network and spatial convolutional attention. Electronics, 2021. 10(19): p. 2444.
33. Shaukat, K., S. Luo, and V. Varadharajan, A novel deep learning-based approach for malware detection. Engineering Applications of Artificial Intelligence, 2023. 122: p. 106030.
34. Zhu, H., et al., An effective end-to-end android malware detection method. Expert Systems with Applications, 2023. 218: p. 119593.
35. Duraibi, S., Enhanced Image-Based Malware Classification using Snake Optimization Algorithm with Deep Convolutional Neural Network. IEEE Access, 2024.
36. Awotunde, J.B., et al., A multi-level random forest model-based intrusion detection using fuzzy inference system for internet of things networks. International Journal of Computational Intelligence Systems, 2023. 16(1): p. 31.
37. Ap, H. and K. K, Secure-MQTT: an efficient fuzzy logic-based approach to detect DoS attack in MQTT protocol for internet of things. EURASIP Journal on Wireless Communications and Networking, 2019. 2019(1): p. 90.
38. Abdel-Basset, M., et al., A risk assessment model for cyber-physical water and wastewater systems: Towards sustainable development. Sustainability, 2022. 14(8): p. 4480.
39. Abdel-Basset, M., et al., A security-by-design decision-making model for risk management in autonomous vehicles. IEEE Access, 2021. 9: p. 107657-107679.

40. Özyurt, F., et al., Brain tumor detection based on Convolutional Neural Network with neutrosophic expert maximum fuzzy sure entropy. Measurement, 2019. 147: p. 106830.
41. Guo, Y. and H.-D. Cheng, New neutrosophic approach to image segmentation. Pattern Recognition, 2009. 42(5): p. 587-595.
42. Kumar, S. and B. Janet, DTMIC: Deep transfer learning for malware image classification. Journal of Information Security and Applications, 2022. 64: p. 103063.
43. Simonyan, K. and A. Zisserman, Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
44. Sandler, M., et al. Mobilenetv2: Inverted residuals and linear bottlenecks. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
45. Huang, G., et al. Densely connected convolutional networks. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
46. Chollet, F. Xception: Deep learning with depthwise separable convolutions. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
47. Howard, A.G., Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.
48. Bozkir, A.S., A.O. Cankaya, and M. Aydos. Utilization and comparision of convolutional neural networks in malware recognition. in 2019 27th signal processing and communications applications conference (SIU). 2019. IEEE.

Received: Mar 1, 2024. Accepted: July 15, 2024